Homomorphic Encryption and the BGN Cryptosystem

David Mandell Freeman

November 18, 2011

1 Homomorphic Encryption

Let's start by considering ElGamal encryption on elliptic curves:

- $\mathsf{Gen}()$: Choose an elliptic curve $E/\mathbb{F}_p$ with a point P of prime order n, and an integer $s \xleftarrow{\mathrm{R}} [1, n]$. Output $\mathsf{pk} = (P, Q = [s]P)$ and $\mathsf{sk} = s$.
- $\mathsf{Enc}(\mathsf{pk}, M)$: Let $\mathsf{pk} = (P, Q)$ and interpret M as a point on $E(\mathbb{F}_p)$. Choose $r \xleftarrow{\mathrm{R}} [1, n]$ and output $C = ([r]P, M + [r]Q)$.
- $\mathsf{Dec}(C, \mathsf{sk})$. Let $\mathsf{sk} = s$ and $C = (C_1, C_2)$. Output $M' = C_2 - [s]C_1$.

Suppose we have two messages $M, M' \in E(\mathbb{F}_p)$ and we compute the encryptions $C = \mathsf{Enc}(\mathsf{pk}, M) = ([r]P, M + [r]Q)$ and $C' = \mathsf{Enc}(\mathsf{pk}, M') = ([r']P, M' + [r']Q)$. Then $C + C'$ (added componentwise) is equal to $([r + r']P, (M + M') + [r + r']Q)$ — so this is an encryption of $M + M'$. Thus given two encryptions of messages, we can obtain an encryption of the sum of the messages *without decrypting*. (Indeed, the system remains semantically secure.) We say the system is "homomorphic": encryption computes a group homomorphism from the message space to the ciphertext space.

When M and M' are points on an elliptic curve, adding the two messages might not make much sense in terms of the underlying meaning of the messages. It would be better if M and M' were integers. For example, if M and M' represent salaries of Stanford professors, then adding them gives the sum of the salaries (and subsequently dividing by 2 gives the average). So here's a variant of ElGamal that encrypts integers:

- $\mathsf{Gen}()$: Choose an elliptic curve $E/\mathbb{F}_p$ with a point P of prime order n, and an integer $s \xleftarrow{\mathrm{R}} [1, n]$. Output $\mathsf{pk} = (P, Q = [s]P)$ and $\mathsf{sk} = s$.
- $\mathsf{Enc}(\mathsf{pk}, M)$: Let $\mathsf{pk} = (P, Q)$ and interpret M as an integer. Choose $r \xleftarrow{\mathrm{R}} [1, n]$ and output $C = ([r]P, [M]P + [r]Q)$.
- $\mathsf{Dec}(C, \mathsf{sk})$. Let $\mathsf{sk} = s$ and $C = (C_1, C_2)$. Compute $C_2 - [s]C_1$ and output $\log_P(T)$.

Note that to decrypt we have to compute a discrete logarithm, which is hard in general. However, if we restrict M to be in some small range — say, positive integers less than some bound B (say $B = 10^9$; we can probably assume no Stanford professor makes a billion dollars), then the discrete log can be quickly computed in time $O(B)$ simply by trying all of the values of $[i]P$ for $i = 1, \ldots, B$. So if the message space is small then decryption is efficient. (Using Pollard's "kangaroo method" we can actually decrypt in $O(\sqrt{B})$ time; see [3] for details.)

Furthermore, this system has the homomorphic property we want; adding two ciphertexts $C = \mathsf{Enc}(\mathsf{pk}, M)$ and $C' = \mathsf{Enc}(\mathsf{pk}, M')$ componentwise now gives an encryption of the integer $M + M'$. We can even apply this operation many times, as long as the total sum doesn't get any bigger than B. So given encryptions of a list of salaries, we could compute an encryption of the sum of all the salaries. We say that this system is "additively homomorphic."

This brings up another issue, related to privacy, which we will describe informally. Suppose that we want to ensure that whoever decrypts the sum of the salaries obtains the sum *only;* i.e., he gets no information about any of the individual salaries. One way to enforce this property is to require that the ciphertext after the sum operation comes from the same distribution as a fresh encryption of the sum itself. To achieve this we "rerandomize" the ciphertext. Namely, we define an algorithm for our ElGamal variant:

- $\mathsf{Add}(\mathsf{pk}, C, C')$: Write $C = (C_1, C_2)$ and $C' = (C'_1, C'_2)$. Choose $t \xleftarrow{\mathrm{R}} [1, n]$ and output $(C_1 + C'_1 + [t]P, C_2 + C'_2 + [t]Q)$.

Thus if the inputs to Add encrypt M and M' with randomness r and r' respectively, the output encrypts $M + M'$ with randomness $r + r' + t$. Since t is independent of r and r', this output is distributed exactly like a fresh encryption of $M + M'$.

In conclusion, we can use our ElGamal variant to compute sums of (small) encrypted data, with the output indistinguishable from a fresh encryption of the sum.

We leave it as an exercise to show that using standard ElGamal encryption in the multiplicative group $\mathbb{F}_p^*$, we can compute *products* of encrypted data. This system is "multiplicatively homomorphic." However, since security relies on the DDH assumption, we need to work in a subgroup of $\mathbb{F}_p^*$ of prime order n. So messages must have order n in $\mathbb{F}_p^*$, and the set of such messages does not have any nice structure — if we want to encrypt a specific integer between 1 and $p - 1$, chances are at most 1 in 2 that the integer has order n (this is if $p = 2n + 1$).

1.1 Fully homomorphic encryption

These observations about ElGamal were made long ago, and there are other other cryptosystems with homomorphic properties. However, until recently all such systems had limitations: most were either additively *or* multiplicatively homomorphic; the few that had both properties required the ciphertext size to grow with the number of operations. (The size of the ElGamal ciphertext remains constant as you perform Add operations.) It was a major open problem to construct an encryption scheme that had both additive and multiplicative homomorphisms. Such a system is called *fully homomorphic*.

What could one do with fully homomorphic encryption? Since addition and multiplication make up a "complete" set of operations (in a sense that we won't make precise), the answer is that one can perform *arbitrary computations on encrypted data.* So for example, I could perform an encrypted Google search: encrypt my input, send it to Google, who performs (many, many) additions and multiplications on my ciphertext and returns to me an answer which I decrypt. This application made fully homomorphic encryption a "holy grail" of cryptography for more than 30 years.

The problem of fully homorphic encryption was solved here at Stanford in 2009, by Craig Gentry in his thesis [4]. Gentry's system uses lattices and algebraic number theory and is outside the scope of this course. However, before Gentry a small but significant advance was made by Boneh, Goh, and Nissim, using pairings on elliptic curves.

2 The BGN Cryptosystem

The cryptosystem devised by Boneh, Goh, and Nissim [1] was the first to allow both additions and multiplications with a constant-size ciphertext. There is a catch, however: while the additive property is the same as for the ElGamal variant, only *one* multiplication is permitted. The system is thus called "somewhat homomorphic."

One of the key ideas in the BGN system is to use elliptic curve groups whose order is a composite number n that is hard to factor. (In all previous systems we required the group order to be prime.) We can generate such a curve as follows: pick two secret primes q and r, and publish $n = qr$. Then find a small integer ℓ such that $4\ell n - 1$ is a prime p, and let E be the elliptic curve $y^2 = x^3 + x$ over $\mathbb{F}_p$. Since $p \equiv 3 \pmod 4$ the curve E is supersingular with $\#E(\mathbb{F}_p) = p + 1 = 4\ell n$, and thus there is a point $P \in E(\mathbb{F}_p)$ of order n. (We can compute such a point by choosing a random $P' \in E(\mathbb{F}_p)$ and setting $P = [4\ell]P'$.)

We now describe the system:

- $\mathsf{Gen}()$: Choose large primes q, r and set $n = qr$. Find a supersingular elliptic curve $E/\mathbb{F}_p$ with a point P of order n as described above, and let $\mathbb{G} = \langle P \rangle$. Choose $Q' \xleftarrow{\text{R}} \mathbb{G} \setminus \{\infty\}$ and set $Q = [r]Q'$; then Q has order q.[1] Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mu_n \subset \mathbb{F}_{p^2}$ be the modified Weil pairing (constructed from the Weil pairing using a distortion map).

  Output the public key $\mathsf{pk} = (E, \hat{e}, n, P, Q)$ and the secret key $\mathsf{sk} = q$.

- $\mathsf{Enc}(\mathsf{pk}, m)$: Choose $t \xleftarrow{\text{R}} [1, n]$ and output $C = [m]P + [t]Q$.

- $\mathsf{Dec}(\mathsf{sk}, C)$: Compute $\tilde{P} = [q]P$ and $\tilde{C} = [q]C$, and output $m' = \log_{\tilde{P}} \tilde{C}$.

Decryption is correct since if $C = [m]P + [t]Q$, then $\tilde{C} = [mq]P + [qt]Q = [mq]P = [m]\tilde{P}$. Note that for efficient decryption we require the message space to be small as in the ElGamal variant.

We now describe how to add encrypted messages and perform one multiplication.

[1] It is extremely unlikely that $Q = \infty$, so we can either ignore this case or say that if it happens we start over.

- $\mathsf{Add}(\mathsf{pk}, C_1, C_2)$: Choose $t' \xleftarrow{\mathrm{R}} [1, n]$ and output $C' = C_1 + C_2 + [t']Q \in \mathbb{G}$.
- $\mathsf{Mult}(\mathsf{pk}, C_1, C_2)$: Choose $u \xleftarrow{\mathrm{R}} [1, n]$ and output $D = \hat{e}(C_1, C_2) \cdot e(Q, Q)^u \in \mu_n$.

It is easy to see that if $C_1 = [m_1]P + [t_1]Q$ is an encryption of m_1 and $C_2 = [m_2]P + [t_2]Q$ is an encryption of m_2, then $\mathsf{Add}(\mathsf{pk}, C_1, C_2)$ is a (rerandomized) encryption of $m_1 + m_2$. With the same setup, we have

$$\begin{aligned} \mathsf{Mult}(\mathsf{pk}, C_1, C_2) &= \hat{e}([m_1]P + [t_1]Q, [m_2]P + [t_2]Q) \cdot \hat{e}(Q, Q)^u \\ &= \hat{e}(P, P)^{m_1 m_2} \hat{e}(P, Q)^{m_1 t_2 + t_1 m_2} \hat{e}(Q, Q)^{t_1 t_2 + u}. \end{aligned}$$

By the nondegeneracy of the pairing, $g = \hat{e}(P, P)$ is an element of $\mathbb{F}_{p^2}$ of order n and $h = \hat{e}(Q, Q)$ is an element of $\mathbb{F}_{p^2}$ of order q. Furthermore, $\hat{e}(P, Q)^q = \hat{e}(P, [q]Q) = 1$, so $\hat{e}(P, Q)$ has order q as well and is equal to h^a for some a. Thus we have

$$D = \mathsf{Mult}(\mathsf{pk}, C_1, C_2) = g^{m_1 m_2} h^{u'}$$

for some integer u'. Since h has order q, raising to the power q allows us to decrypt D:

- $\mathsf{Dec}'(\mathsf{sk}, D)$: Compute $\tilde{g} = g^q$ and $\tilde{D} = D^q$, and output $\log_{\tilde{g}} \tilde{D}$.

Furthermore, we can add ciphertexts D_1, D_2 in μ_n:

- $\mathsf{Add}'(\mathsf{pk}, D_1, D_2)$: Choose $u \xleftarrow{\mathrm{R}} [1, n]$ and output $D' = D_1 \cdot D_2 \cdot h^u$.

To obtain an encryption of m in μ_n we could encrypt m directly as $g^m h^u$ or encrypt m in $\mathbb{G}$ using Enc and pair the ciphertext with an encryption of 1.

Note that the Mult algorithm moves ciphertexts from $E(\mathbb{F}_p)$ to μ_n. Since there is no pairing on μ_n (or more generally, on $\mathbb{F}_{p^2}^*$), we cannot perform any more multiplication operations.

In conclusion, using the BGN system we can encrypt messages $m_1, \ldots, m_\ell$ in the group $\mathbb{G}$ and use the Add, Mult, and Add' algorithms to evaluate any ℓ-variable quadratic polynomial in the m_i.

2.1 Security

Clearly, if we can factor the group order n in the BGN cryptosystem, then we know the secret key q and the system is broken. Also, if we can compute discrete logarithms in the group $\mathbb{G}$, then we can compute $r = \log_P(Q)$ and $q = n/r$. So necessary conditions for security are that factoring n is hard and computing discrete logarithms in $\mathbb{G}$ is hard.

It turns out that a sufficient condition for security is the following: given $R \in \mathbb{G}$, it is hard to determine whether R has order q. Or in other words, it is hard to determine whether R is in the cyclic subgroup of $E(\mathbb{F}_p)$ generated by Q. This is the *subgroup decision problem*, which we now define formally.

Definition 1. *Let $\mathbb{G} \subset E(\mathbb{F}_p)$ be a group of order $n = qr$ with q, r prime. Define*

$$\text{SD-Adv}(\mathcal{A}, \mathbb{G}) = \left| \Pr[\mathcal{A}(P, Q) = 1 : P, Q \xleftarrow{\text{R}} \mathbb{G}] - \Pr[\mathcal{A}(P, Q = 1) : P, Q' \xleftarrow{\text{R}} \mathbb{G},\ Q = [r]Q'] \right|.$$

We say that $\mathbb{G}$ satisfies the ϵ-subgroup decision assumption *if for all efficient adversaries $\mathcal{A}$,* $\text{SD-Adv}(\mathcal{A}, \mathbb{G}) < \epsilon$.

In other words, a challenge for the subgroup decision problem consists of a pair (P, Q) where P has order n and Q has order either n or q, and a subgroup decision adversary tries to determine which of these is the case. It is clear that if we can factor n then we can solve the subgroup decision problem by simply testing where $[q]Q = \infty$.

We can now prove security of the BGN cryptosystem.

Theorem 1. *If $\mathbb{G}$ satisfies the ϵ-subgroup decision assumption, then the BGN cryptosystem is 2ϵ-semantically secure.*

Proof. We use the "real-or-random" definition of semantic security. Suppose the BGN cryptosystem is not 2ϵ-semantically secure. Then there exists an adversary $\mathcal{A}$ that plays the semantic security game such that $\text{SS-Adv}(\mathcal{A}, \text{BGN}) \geq 2\epsilon$. Suppose we are given a subgroup decision challenge (P, Q). We construct an algorithm $\mathcal{B}$ that acts as a semantic security challenger for $\mathcal{A}$ as follows:

- $\mathcal{B}$ gives $\mathcal{A}$ the BGN public key $\mathsf{pk}(P, Q)$.
- When $\mathcal{A}$ submits its message query m, $\mathcal{B}$ chooses $t \xleftarrow{\text{R}} [1, n]$ and responds with the ciphertext $c = [m]P + [t]Q$.
- $\mathcal{B}$ outputs the same as $\mathcal{A}$.

Now let us analyze the output of $\mathcal{B}$.

- When Q has order q, pk is a real BGN public key and c is a real encryption of m.
- When Q has order n, pk is a "bad" BGN public key (since in a real public key Q has order q), and c is a uniformly random element of $\mathbb{G}$.

Thus we have

$$\begin{aligned}
\text{SD-Adv}(\mathcal{B}, \mathbb{G}) &= \left| \Pr[\mathcal{A}(P, Q) = 1 : \text{real } \mathsf{pk},\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] - \Pr[\mathcal{A}(P, Q) = 1 : \text{bad } \mathsf{pk},\ c \xleftarrow{\text{R}} \mathbb{G}] \right| \\
&= \Big| \Pr[\mathcal{A}(P, Q) = 1 : \text{real } \mathsf{pk},\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] \\
&\qquad - \Pr[\mathcal{A}(P, Q) = 1 : \text{real } \mathsf{pk},\ c \xleftarrow{\text{R}} \mathbb{G}] + \Pr[\mathcal{A}(P, Q) = 1 : \text{real } \mathsf{pk},\ c \xleftarrow{\text{R}} \mathbb{G}] \\
&\qquad\quad - \Pr[\mathcal{A}(P, Q) = 1 : \text{bad } \mathsf{pk},\ c \xleftarrow{\text{R}} \mathbb{G}] \Big|
\end{aligned}$$

Let α, β, γ be the three probabilities in the last expression, so SD-Adv$(\mathcal{B}, \mathbb{G}) = \big|\alpha - \beta + \beta - \gamma\big|$. Since the encryption of a random message m (under a real public key) is a uniformly random element of $\mathbb{G}$,

$$\big|\alpha - \beta\big| = \text{SS-Adv}(\mathcal{A}, \text{BGN}) \geq 2\epsilon.$$

It now follows from the triangle inequality that either SD-Adv$(\mathcal{B}, \mathbb{G}) \geq \epsilon$ or $\big|\beta - \gamma\big| \geq \epsilon$. If the first case holds, then we are done. In the second case, we construct an algorithm $\mathcal{B}'$ that take a real public key (P, Q) and acts the same as $\mathcal{B}$, except that $\mathcal{B}'$ always gives $\mathcal{A}$ a uniformly random ciphertext c. Then SD-Adv$(\mathcal{B}', \mathbb{G}) = \big|\beta - \gamma\big| \geq \epsilon$, and again we are done.

We conclude that if $\mathcal{A}$ breaks the semantic security of the BGN system, then either $\mathcal{B}$ or $\mathcal{B}'$ can break the subgroup decision assumption, proving the theorem. □

2.2 An application

Let $\vec{v}_i^{(\ell)}$ be the unit vector

$$\vec{v}_i^{(\ell)} = \underbrace{(0, \ldots, 0, \overbrace{1}^{i\text{th place}}, 0, \ldots, 0)}_{\ell \text{ entries}}.$$

Suppose we want to encrypt $\vec{v}_i^{(\ell)}$. The naive way to do this is to encrypt each entry, giving a vector of ℓ ciphertexts. Using the BGN system, we can shorten this length significantly.

Assume for simplicity that $\ell = m^2$ for some integer m. Write $i = a + bm$ for integers $1 \leq a, b \leq m$. Now form the BGN encryptions of $\vec{v}_a^{(m)}$ and $\vec{v}_b^{(m)}$. That is,

$$\vec{C} = (C_1, \ldots, C_m) \qquad C_j = \begin{cases} \mathsf{Enc}(\mathsf{pk}, 0) & \text{if } j \neq a \\ \mathsf{Enc}(\mathsf{pk}, 1) & \text{if } j = a \end{cases}$$

$$\vec{C}' = (C'_1, \ldots, C'_m) \qquad C_j = \begin{cases} \mathsf{Enc}(\mathsf{pk}, 0) & \text{if } j \neq b \\ \mathsf{Enc}(\mathsf{pk}, 1) & \text{if } j = b \end{cases}$$

We let $(\vec{C}, \vec{C}')$ be the encryption of $\vec{v}_i^{(\ell)}$; note that this consists of $2m = 2\sqrt{\ell}$ ciphertexts, which is much better than ℓ ciphertexts using the naive method.

How does the secret key holder recover $\vec{v}_i^{(\ell)}$? First he forms the ciphertext vector $\vec{D} = (D_1, \ldots, D_\ell)$, where D_k is computed by writing $k = a' + b'm$ and letting $D_k = \mathsf{Mult}(\mathsf{pk}, C_{a'}, C'_{b'})$. Then he uses the Dec' algorithm to decrypt the entries of $\vec{D}$. The multiplicative property of the BGN system ensures that the only entry that decrypts to 1 is the entry with $k = i$.

This encryption of a unit vector has applications to "private information retrieval"; details are in the BGN paper [1, Section 4].

2.3 A note on efficiency

In most pairing-based cryptosystems, we require the discrete logarithm to be hard on $E(\mathbb{F}_p)$ and in the finite field $\mathbb{F}_{p^k}$ in which the pairing takes values. Since there are no subexponential-time

discrete log algorithms on elliptic curves but there are such algorithms in $\mathbb{F}_{p^k}$, this means that we can use curves over fields as small as 160 bits as long as p^k has at least 1024 bits. (Here k is the "embedding degree.") Supersingular curves over $\mathbb{F}_p$ have $k = 2$ and thus require p to have 512 bits; much work has gone into constructing curves with larger k, which allow for smaller p.

In the BGN system, on the other hand, we require factoring the group order n to be hard. Since there are subexponential-time factoring algorithms, we need n to be at least 1024 bits, so for efficiency after pairing we want k to be as *small* as possible. Thus supersingular curves over $\mathbb{F}_p$ are optimal in terms of security and efficiency for the BGN system. For more details, see [2, Section 8.4].

References

[1] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," TCC 2005, http://crypto.stanford.edu/~dabo/pubs/papers/2dnf.pdf.

[2] D. Freeman, M. Scott, and E. Teske, "A taxonomy of pairing-friendly elliptic curves," http://eprint.iacr.org/2006/372.

[3] S. Galbraith, "Factoring and discrete logarithms using pseudorandom walks," http://www.math.auckland.ac.nz/~sgal018/crypto-book/ch15.pdf.

[4] C. Gentry, "Fully homomorphic encryption using ideal lattices," http://crypto.stanford.edu/craig.